Estimating Software Effort with Minimum Features using Neural Functional Approximation

Pichai Jodpimai, Peraphon Sophatsathit, and Chidchanok Lursinsap
Advanced Virtual and Intelligent Computing (AVIC) Center
Department of Mathematics, Faculty of Science, Chulalongkorn University
Bangkok, Thailand
E-mail: pichai.j@student.chula.ac.th, peraphon.s@chula.ac.th, lchidcha@pioneer.netserv.chula.ac.th

Abstract— **The aim of this study is to improve software effort estimation by incorporating straightforward mathematical principles and artificial neural network technique. Our process consists of three major steps. The first step concerns data preparation from each considered database. The second step is to reduce the number of given features by considering only those relevant ones. The final step is to transform the problem of estimating software effort to the problems of classification and functional approximation by using a feedforward neural network. Experimental data are taken from well-known public domains. The results are systematically compared with related prior works using only a few features so obtained, yet demonstrate that the proposed model yields satisfactory estimation accuracy based on MMRE and PRED measures.**

Keywords- ***Software Effort Estimation; Artificial Neural Networks; Functional Approximation; MMRE; PRED***

I. INTRODUCTION

Software effort estimation is the process in planning stage of software development life cycle for predicting the software effort to estimate software costs required [1]. An estimation precision of software project cost is important for software project management. A numbers of techniques have been introduced in software development effort estimation such as regression analysis, statistical model, genetic algorithm, fuzzy, fuzzy-neuro systems, and artificial neural networks. An artificial neural network is a computational approach to model complex relationships between independent and dependent features. However, the problems still significantly challenge researchers and experts to improve estimation accuracy. One predominant difficulty that estimation models must reckon with is the number of cost drivers involved in the effort prediction process. Consequently, the objectives of this research are to enhance software effort estimation accuracy and to reduce the number of features. The proposed model compares its estimation with the actual value and employs MMRE and PRED measures to gauge the estimation accuracy, all of which are based on smaller number of candidate features than what originally have available.

The organization of this paper is as follows. Section 2 discusses literature review on software effort estimation models. Section 3 describes details of the proposed model encompassing data source and methodology. Section 4 explains some note worthy implications of the experiments. The conclusion and future work are discussed in the last section.

II. RELATED WORK

The early techniques for software effort estimation were typically based on statistics and regression analysis. Boehm's COnstructive COst MOdel (COCOMO) [2] and Putnam's Software LIfecycle Management (SLIM) [3] are early well-known software effort estimation models. In the mean time, many new approaches have been proposed. In analogy-based estimation, Li, et al [4] introduced the methodology for selection of proper projects using genetic algorithm to improve the performance of analogy-based effort prediction. Chiu and Huang [5] considered improving analogy-based effort estimation with the help of genetic algorithm for adjusting reuse effort by measuring similarity distances between pairs of projects using Euclidean, Manhattan, and Minkowski methods. Huang, et al [6] investigated the potential means to improve the accuracy of effort estimation, where grey relational analysis and genetic algorithm were used to handle similarity measures of complex relations and found appropriate weight in each effort driver, respectively. Lefley and Shepperd [7] indicated that genetic programming yielded good accuracy of software effort estimations based on a management consultancy organization. Kumar, et al [8] applied wavelet neural network (WNN) for software effort prediction with the help of Morlet and Gaussian transfer functions. In addition, they proposed a threshold acceptance training algorithm for wavelet neural network (TAWNN). Tadayon [9] investigated the use of expert judgment, artificial neural networks, and COCOMO II model for software effort prediction. Rao, et al [10] proposed Functional Link Artificial Neural Network (FLANN) to reduce the computational complexity suitable for on-line applications. Pedrycz, et al [11] introduced a granular model of software effort estimation designed based on experimentally developed information granules. It was concerned with the development of software effort estimation models using fuzzy sets. Huang and Chiu [12] proposed a fuzzy neural network for software effort estimations by applying artificial neural network to fuzzy inference processes. Recently, researchers try to take other technique mix into the original models to improve effort prediction accuracy.

Another important aspect of effort estimation is feature subset selection method. Chen, et al [13] suggested effort improvement estimation using WRAPPER technique to select the feature subset. Koch and Mitlohner [14] employed the concept of social choice for software project effort estimation, substituting the voters by software project attributes. Huang and Chiu [15] investigated the effect on estimation accuracy by adopting genetic algorithm for use in analogy-based software effort estimation models to determine the appropriate weighted similarity measures of effort drivers. The approach used three weighted analogy methods, namely, the unequally weighted, the linearly weighted, and the nonlinearly weighted. The linearly weighted deals with a linear equation for each effort driver as the weight of distance measure; the unequally weighted assigns a constant weight to distance measure for each effort driver; and the nonlinearly weighted uses a separate nonlinear equation as a weight of distance measure for each effort driver. Shepperd and Schofield [16] employed analogies to estimate software effort data through the case-base reasoning technique and other prediction approaches. In short, researchers have expended numerous endeavors on effort estimation at the expense of full fledged software project attributes. As such, we propose a technique that attempts to reduce the costly estimation process by utilizing only pertinent software project features, attributes, or cost drivers (depending on the underlying metrics, e.g., FP, KSLOC, KDSI, etc.), but hereafter will be referred to as *features*.

III. Proposed Methodology

Our process consists of three major steps. The first step concerns data preparation from each considered database. The second step is to reduce the number of given features by considering only those relevant ones. The final step is to transform the problem of estimating software effort to the problems of classification and functional approximation by using a feedforward neural network. Detail of each step is described in the sections that follow.

A. Data Preparation and Database Selection

All standard data sets are chosen from available software engineering public domain as follows. First and foremost, COCOMO81 data set [2] was selected, consisting of 63 software projects in various programming languages, such as FORTAN, Pascal, and C. The nature of the projects includes Business, Process control, Human-Machine Interaction, Scientific, Support, and System. To construct the model, the original 16 independent features and one dependent feature from the data set to accommodate COCOMO81 Equation are adopted and computed as shown in (1).

$$Effort = a * (KSLOC^{b}) * (\prod_{i=1}^{15} EM_{i}). \qquad (1)$$

where Effort is the estimated effort measured in person-months; *a* and *b* are specific parameters representing three software development modes, namely, Organic, Semi-detached, and Embedded; KSLOC is the thousands of lines of code in the program excluding blank and comment lines (some literatures use KDSI whose meaning is slightly different, but is considered interchangeable with KSLOC in the context of this work as the focus is on more important issues). The value of KSLOC is estimated directly or derived from function point analysis (FPA); and EM is the effort multiplier measured from project's environmental factors, usually referred to effort drivers.

There are 60 projects from 1980 to 1990 in NASA60 data set [17] and 93 projects from 1971 to 1987 in NASA93 [18]. Both data sets are in COCOMO81 format collected from different NASA centers published in PRedictOr Models In Software Engineering (PROMISE).

Albrecht data set [19], [20] consists of 24 software projects from IBM Data Processing Services (IBMDPS) organization developed in third generation languages. Of the 24 projects, 18 were written in COBOL, 4 in PL1, and 2 in DMS language. We employed 6 features in the model, encompassing 5 independent features which include number of inputs, number of outputs, number of inquiries, number of master files, and adjustment factor, and one dependent feature which is the actual effort measured in work-hours. Meanwhile, we exclude source lines of code that appears in this data set since it is not a parameter for calculating function point using Albrecht model.

CF data set [21] has 21 historical projects derived from Canadian Financial organization, using the FPA rules of the International Function Point Users Group (IFPUG) standards. We adopted 7 features which include 6 independent features consisting of internal logical files, external interface files, external inputs, external outputs, external inquiries, and adjustment factor, and one dependent feature which is the actual effort measured in the number of actual working days spent to finish the project.

Desharnais data set [22] contains 81 software projects. In this study, four projects were excluded because the data were incomplete. There are 10 independent features and one dependent feature, which is the actual effort measured in person-hours, constituting the model.

Table I-IV explain each feature from the above data sets. Table I describes details of features from COCOMO81, NASA 60, and NASA93 datasets. Feature 1 to 16 are independent features, and feature 17 is the dependent feature. Table II-IV describes the same feature pattern from Albrecht, CF, and Desharnais data sets, i.e., all but last feature are independent features, respectively. These independent features are used as inputs for the neural network, while the dependent feature represents the output. Table V summarizes the number of projects collected, the minimum and maximum values of software effort in each data set.

TABLE I. DESCRIPTION OF FEATURES FROM COCOMO81, NASA60, AND NASA93 DATA SETS

No.	Feature	Description
1	RELY	Required Reliability
2	DATA	Database Size
3	CPLX	Product Complexity
4	TIME	Execution Time Constraint
5	STOR	Main Storage Constraint
6	VIRT	Virtual Machine Volatility
7	TURN	Computer Turnaround Time
8	ACAP	Analyst Capability
9	AEXP	Application Experience
10	PCAP	Programmer Capability
11	VEXP	Virtual Machine Experience
12	LEXP	Programming Language Experience
13	MODP	Modern Programming Practices
14	TOOL	Use of Software Tools
15	SCED	Required Development Schedule
16	KSLOC	Kilo Source Lines of Code
17	EFFORT	Actual effort is measured in person-months

TABLE II. DESCRIPTION OF FEATURES FROM ALBRECHT DATA SET

No.	Feature	Description
1	IN	Number of inputs
2	OUT	Number of outputs
3	FILE	Number of master files
4	INQ	Number of inquiries
5	ADJ	Adjust Function Point
6	EFFORT	Actual effort is measured in work-hours

TABLE III. DESCRIPTION OF FEATURES FROM CF DATA SET

No.	Feature	Description
1	ILF	Internal Logical Files
2	EIF	External Interface Files
3	EI	External Inputs
4	EO	External Outputs
5	EQ	External Inquiries
6	AF	Adjustment Factor
7	EFFORT	Actual effort measured in real workingdays

TABLE IV. DESCRIPTION OF FEATURES FROM DESHARNAIS DATA SET

No.	Feature	Description
1	TeamExp	Experience of project team in years
2	ManagerExp	Experience of project manager in years
3	YearEnd	Year of finished project
4	Length	-
5	Transactions	Number of transactions processed
6	Entities	Number of entities in the systems data model
7	PointsAdjust	Adjusted Function Points
8	Envergure	-
9	PointsNonAdjust	Unadjusted function point
10	Language	Language {1,2,3}
11	EFFORT	Actual effort is measured in person-hours

TABLE V. DATA SET SUMMARY

Data Set	No. of projects	Minimum value of effort	Maximum value of effort
COCOMO81	63	5.90	11,400
NASA60	60	8.40	3,240
NASA93	93	8.40	8,211
Albrecht	24	0.50	105.20
CF	21	52	544
Desharnais	77	546	23,940

B. Feature Selection

Our effort estimation is based on the assumption that there exist some relevant software features out of all features in each data set that actually exert significant affect on software effort estimation. This assumption leads to the problem of how to select those relevant features. The solution to this problem depends on the observation that estimating any software effort from a set of given features is similar to developing a function to compute the value of software effort using the given features as its variables. Let $x_i^{(k)}$ for $1 \leq i \leq m$ be the selected feature i of project k and m be the number of selected features considered in project k. Thus, the software effort of project k, denoted by $E^{(k)}$, with those given features $\{x_i^{(k)} \mid 1 \leq i \leq m\}$ can be related and written in forms of a mathematical function

$$E^{(k)} = f\,(x_1^{(k)}, x_2^{(k)}, \dots , x_m^{(k)}). \tag{2}$$

Since this function is a multivariate function, it is rather difficult to create the function by using a multivariate polynomial regression technique with some degree. In this paper, a neural network is deployed for functional approximation because of its efficiency and simplicity. The output of each hidden neuron as well as output neuron is generated by a logistic function. Thus, it is rather straight forward to derive the function of software effort from the neural network. Suppose there are two input features, two hidden neurons, and one output neuron in a neural network shown in Fig. 1. All inputs are fully connected with all hidden neurons and all hidden neurons are fully connected with the output neuron. Let $x_i^{(k)}$ for $1 \le i \le 2$ be the input of pattern k. Let h_1, h_2, h_3 be the biases of hidden neurons 1, 2, and output neuron, respectively. Weights of hidden neuron 1 and 2 are w_{11}, w_{12}, w_{21}, w_{22}, and of output neuron are w_{31}, w_{32}, respectively. The outputs are denoted by O_1 and O_2. The function implemented by this network has the following form.

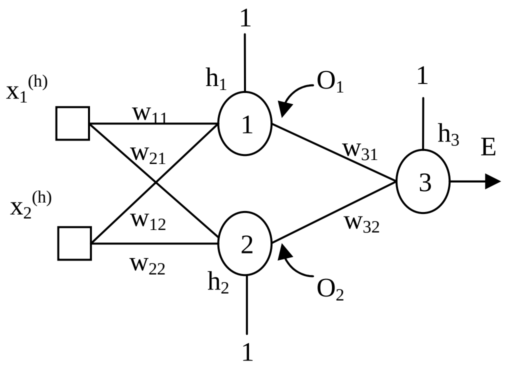

Figure 1. An example of Neural network model.

$$O_1 = \frac{1}{1+e^{-(x_1^{(k)} \cdot w_{11} + x_2^{(k)} \cdot w_{12} + h_1)}} . \tag{3}$$

$$O_2 = \frac{1}{1+e^{-(x_1^{(k)} \cdot w_{21} + x_2^{(k)} \cdot w_{22} + h_2)}} . \tag{4}$$

$$E = \frac{1}{1+e^{-(O_1 \cdot w_{31} + O_2 \cdot w_{32} + h_3)}} . \tag{5}$$

To select those relevant features, the relationship between the software effort $E^{(k)}$ and the relevant features must retain the property of a mathematical function. This implies that a selected set of features must spend only one value of software effort. If the selected set of features can spend more than one software efforts, the property of a mathematical function will no long hold, thereby the function cannot be created. Since the maximum number features in the benchmark data sets is only 16, it is feasible to try all possible combinations of the selected features. Let $I = \{1, 2, 3, \ldots, m\}$ be a set of feature indices. Suppose there are p projects in our data set.

Feature Selecting Algorithm

```
1.   Let $S$ be an empty set.
2.   **for** each $j \in [1,m]$ **do**
3.       Let $F_j$ be an empty set.
4.       Generate all possible combinations of $j$ feature
         indices from set $I$ and make each combination as
         a set.
5.       Put each combination set in $F_j$.
6.       **for** each feature combination index set $s \in F_j$ **do**
7.           **for** each $i \in [1, p-1]$ **do**
8.               **for** each $k \in [i, p]$ **do**
9.                   **if** project $i$ and project $k$ have exactly the
                       same feature values under feature
                       combination index set $s$ but different
                       $E^{(i)}$ and $E^{(k)}$ **then** discard set $s$ and exit
                       for loop
10.              **endfor**
11.              Let $S = S \cup s$
12.          **endfor**
13.      **endfor**
14.  **endfor**
```

A relation between the selected features and their software efforts can be a function if and only if the relation is many-to-one or one-to-one. Based on this functional relation, the algorithm filters all possible functional relations under different numbers of selected features and selects only those relations with minimum number of selected features for creating the function of software effort. Table VI shows an example of the how algorithm works.

TABLE VI. AN EXAMPLE OF GIVEN DATA SET FOR EXPLAINING THE ALGORITHM

No. of Projects	Feature 1	Feature 2	Feature 3	Software effort
1	1	2	3	5
2	2	1	2	3
3	1	3	2	4
4	2	2	1	6

From Table VI, it is obvious that if only one feature is selected, i.e. feature 1, feature 2, or feature 3, it is impossible to create a function of software effort with the selected feature as it variable since the same feature gives different software efforts. However, if at least two features are selected, namely features {1,2}, {1,3}, {2,3}, or {1,2,3}, it is possible to create a function of software effort based on the selected features. In this case, the minimum number of selected features possible for creating the function of software effort is 2. Thus, either features {1,2}, {1,3}, or {2,3} can be used for this purpose. Table VII shows some combinations of selected features from COCOMOS1 data set.

TABLE VII. SELECTED SOME FEATURES FROM CANDIDATE LIST OF COCOMO81 DATA SET

Candidate List	Feature 1	Feature 2	Feature 3
1	1	3	16
2	1	4	16
3	1	5	16
4	1	6	16

Candidate List	Feature 1	Feature 2	Feature 3
5	1	7	16
6	1	8	16
7	1	9	16
8	1	11	16
9	1	12	16
10	1	13	16
11	1	14	16
12	1	15	16
13	2	4	16
14	2	5	16
15	2	6	16

C. Classification and Functional Approximation

The final step is to create a software effort function as previously discussed. The most efficient and feasible technique is a feedforward neural network (FNN). However, the value of software effort in each project varies from a single digit to five digits. Table V summarizes the minimum and maximum values of software effort in each data set. Such a wide range of values makes the learning convergence and accuracy of estimation for the neural network almost impossible. To resolve these obstacles, all values must be grouped into their corresponding ranges. In our study, the values of software effort of each data set are grouped into two ranges as summarized in Table VIII. The output layer holds the functional approximation of project effort obtained from the FNN estimation.

TABLE VIII. GROUP RANGES OF THE SOFTWARE EFORT VALUES

Data Set	The 1st range	The 2nd range
COCOMO81	5.90-88	98-11,400
NASA60	8.40-117.60	117.61-3,240
NASA93	8.40-600	636-8,211
Albrecht	0.50-11.10	11.80-105.20
CF	52-363	369-544
Desharnais	546-3,472	3,542-23,940

There are two major steps in creating the software effort function. The first step makes use of the FNN to classify the selected features of each project into the aforementioned two ranges. The second step performs the functional approximation for each group. During the classification of software effort value ranges, we set the target of the first range group to be 1 and the target of the second range to be 0. The total number of projects is partitioned into 70% training set and 30% testing set. One neural network is used to carry out this classification. On the other hand, two neural networks are used in the functional approximation step to create software effort functions. The first network is deployed to estimate the software effort values in the first range while the second network is for the second range. The total number of projects is partitioned into two groups according to the range of software effort values. Each group is then divided into 70% training set and 30% testing set.

After creating the software effort function, testing process is performed by predicting the group of software effort value range first. Then, the neural network corresponding to the software value range is used for approximating the software effort value. Fig. 2 illustrates the testing process. Both classification and functional approximation computation processes are realized by the Stuttgart Neural Network Simulator (SNNS) [23].

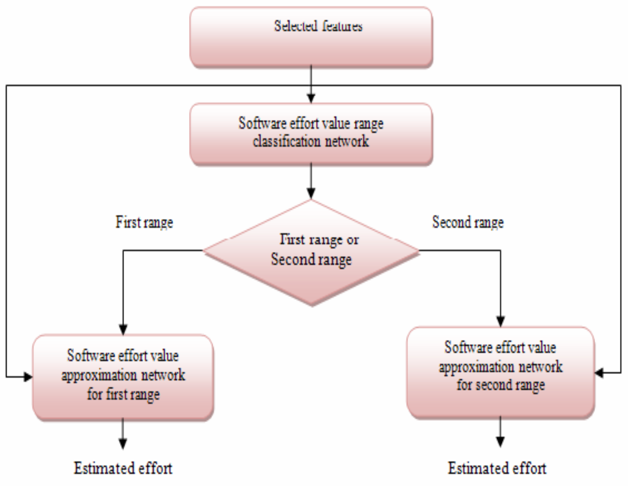
Selected features
Software effort value range classification network
First range
Second range
First range or Second range
Software effort value approximation network for first range
Software effort value approximation network for second range
Estimated effort
Estimated effort

Figure 2. Testing process of the proposed method.

D. Evaluation

To assess the output accuracy of the proposed model, we use Mean Magnitude Relative Error (MMRE) and Percentage Relative Error Deviation within *x* or PRED(x) based on the same basic unit-less value of Magnitude Relative Error (MRE) [24]. The MRE is defined as

$$MRE_i = \frac{|\hat{y}_i - y_i|}{y_i} . \quad (6)$$

where y_i is the actual effort, and $\hat{y}_i$ is the estimated effort, both of which are used in software project *i*. The MMRE and PRED(x) are defined below.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i . \quad (7)$$

where *N* is the total number of software projects. Generally, the value of MMRE smaller than or equal to 0.25 is considered an acceptable level of performance for effort estimation model [24].

$$PRED(x) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 1 & if \quad MRE_i \le x \\ 0 & Otherwise \end{cases} . \quad (8)$$

PRED(x) is the percentage of prediction within *x* percent of the actual value. Normally, the value of *x* is set to 0.25 corresponding to the acceptable level of MMRE. A small MMRE value and large PRED value are preferred as they are apparent from Equation 7 and 8 to attain any reasonable software effort predictions.

In addition to adopting standard MMRE and PRED(x) measures, we compared the estimated effort determined by the proposed model with the actual effort from each data set. The resulting estimation will be described in the next section.

IV. EXPERIMENTAL RESULTS

This section explains the accuracy of average effort estimation by the proposed model, as well as the performance against other models.

A. *Results of effort estimation*

We employed 30% of test data set from COCOMO81, NASA60, NASA93, Albrecht, CF, and Desharnias to measure the estimated efforts in comparison with the actual values. The comparative results are depicted in Table IX, depicting the actual effort value (obtained from the data set), estimated effort value using a full-fledged feature set, and estimation of the proposed model computed by only a few selected features. The set of selected features utilized in the estimation by the proposed model from each data set are RELY, VIRT, and KSLOC (COCOMO81), CPLX, PCAP, and KSLOC (NASA60), DATA, TURN, PCAP, and KSLOC (NASA93), FILE and INQ (Albrecht), EIF and AF (CF), and Entities and PointsNonAdjust (Desharnais) data sets.

TABLE IX. COMPARATIVE RESULTS OF ESTIMATED EFFORT AND ACTUAL EFFORT FOR ALL DATA SETS

Data Set	**Test Set**	**Actual Effort**	**Estimated Effort**			
			Original Model		***Our Model***	
		Average Effort	***No.feature***	***Average Effort***	***No.feature***	***Average Effort***
COCOMO81	18	207.90	16	232.88	3	228.99
NASA60	16	340.19	16	-	3	354.66
NASA93	26	734.03	16	-	4	722.96
Albrecht	6	12.07	5	16.67	2	12.04
CF	6	359.33	6	-	2	353.42
Desharnais	22	5,119.36	10	-	2	4,852.17

Fig. 3-8 depict graphic plots of average estimated effort against average actual effort in all test sets, where x-axis represents software project test sets, y-axis represents actual effort, starred-solid line represents actual effort, and circle-solid line represents estimated effort. For each data set, the estimation computed by the few selected features is compared with the actual value of that set. For example, the plot of COCOMO81 data set of Fig. 3 shows that only the 10th project test set exhibited an unusual estimation burst. The sporadic phenomena were observed in NASA93 and CF data sets as well. Such idiosyncrasies can be attributed from several causes. The predominant ones are (1) duplicated inputs (2) statistically inadequate data points in the data set (3) unusually dispersed data values (as shown in Table V). The latter two causes are the major caveats that hinder the estimation process, thereby the neural network model is unable to adequately capture the significance of dispersion pattern to arrive at any closer prediction. This is the only limitation of the proposed model which is the same limitation for all statistical inference techniques.

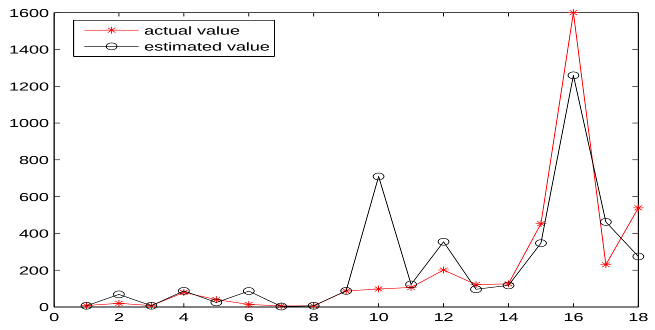

Figure 3. Comparative results of actual and estimated efforts with COCOMO81 data set.

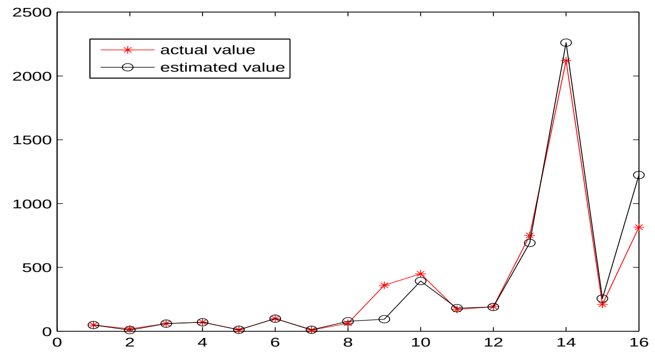

Figure 4. Comparative results of actual and estimated efforts with NASA60 data set.

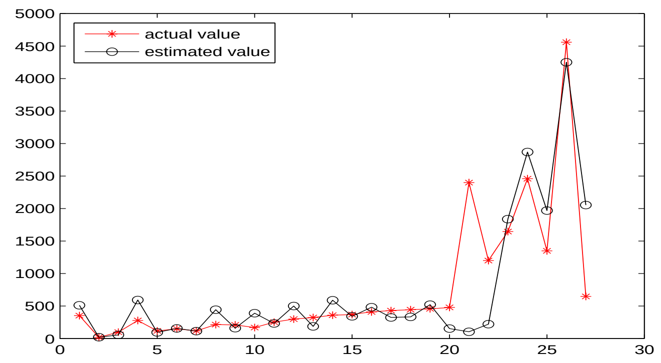

Figure 5. Comparative results of actual and estimated efforts with NASA93 data set.

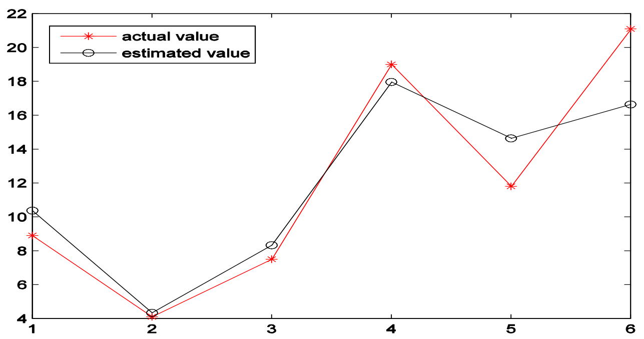

Figure 6. Comparative results of actual and estimated efforts with Albrecht data set.

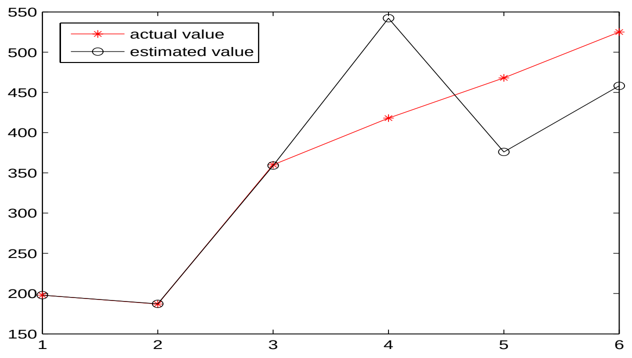

Figure 7. Comparative results of actual and estimated efforts with CF data set.

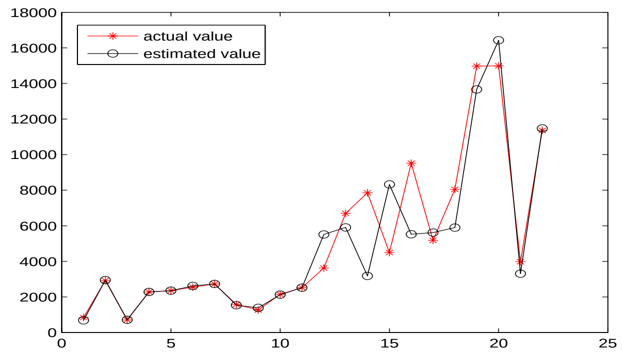

Figure 8. Comparative results of actual and estimated efforts with Desharnais data set.

B. *Comparative results*

Evaluation of standard comparative measures, i.e., MMRE and PRED(0.25), except for feature subset selection [13] reporting PRED(0.30), for the proposed model against other estimation models was carried out. A few well established ones are Grey relational analysis [6], Case based reasoning [6], Classification and regression trees [6], Artificial neural network [6], and Social choice (weighted, CO) [14] in COCOMO81 data set, Feature subset selection [13] in NASA60 data set, Project selection analogy based estimation [4], Adjusted analogy based estimation-Minkowski [5], Grey relation analysis with genetic algorithm [6], Wavelet neural network-Gaussian [8], Social choice (weighted, BO) [14], Unequally weighted analogy [15], Linearly weighted analogy [15], Nonlinearly weighted analogy [15], and Analogy based estimation [16] in Albrecht data set, Wavelet neural network-Morlet [8], Wavelet neural network-Gaussian, and Adjusted analogy based estimation-Euclidean distance [5] in CF data set, Project selection analogy based estimation [4], Artificial neural network [4], Redial basis function [4], Classification and regression trees [4], and Analogy based estimation [16] in Desharnais data set. The comparative results are shown in Table X.

TABLE X. COMPARATIVE RESULTS OF PROPOSED AND SELECTED MODELS BASED ON STANDARD DATA SETS

Model	PRED (0.25)	MMRE
Comparison with COCOMO81 data set		
Proposed model	0.56	1.00
Grey relational analysis with genetic algorithm	0.38	0.69
Case based reasoning	0.12	4.46
Classification and regression trees	0.25	2.44
Artificial neural network	0.11	1.43
Social choice (weighted, CO)	0.46	10.42
Comparison of FSS model with NASA60 data set		
Proposed model	0.81	0.18
Feature subset selection	0.81(0.30)	-
Comparison with Albrecht data set		
Proposed model	1.00	0.14
Project selection analogy based estimation	0.25	0.42
Adjusted analogy based estimation-Minkowski	0.61	0.43
Grey relation analysis with genetic algorithm	0.48	0.32
Wavelet neural network-Gaussian	0.88	0.07
Social choice (weighted, BO)	0.71	0.98
Unequally weighted analogy	0.64	0.44
Linearly weighted analogy	0.70	0.32
Nonlinearly weighted analogy	0.57	0.33
Analogy based estimation	0.33	0.62
Comparison with CF data set		
Proposed model	0.83	0.10
Wavelet neural network-Morlet	0.67	0.20
Wavelet neural network-Gaussian	0.57	0.23
Adjusted analogy based estimation-Euclidean distance	0.43	0.52
Comparison with Desharnais data set		
Proposed model	0.77	0.16
Project selection analogy based estimation	0.33	0.41
Artificial neural network	0.22	0.57

Model	PRED (0.25)	MMRE
Redial basis function	0.33	0.42
Classification and regression trees	0.30	0.52
Analogy based estimation	0.36	0.64

The above results indicate that the proposed model yields the highest PRED(0.25) and near optimal MMRE for every data set.

V. CONCLUSION

This research has demonstrated the proposed neural networks approach that yields high software effort prediction accuracy in comparison with other well established effort estimation models using only a few features. The findings were further reaffirmed by remakable comparative results of the average estimation effort with the corresponding actual values. The implications of the proposed endeavor are seveal folds. First and foremost, the model can serve as a cost-saving project effort estimation means to pinpoint and select only relevant and necessary features. Second, project managers and experts can spend less time to predict software project cost and more time on more important issues. Third, less computation time and effort are required, thereby energy saving mandates can be easily adhered. And last but not least, the proposed approach will lend itself to future work on machine learning for efficient and accurate project cost estimation.

Bearing all of the above contributions in mind, two important future research candidates are (1) the aforementioned limitation about data adequacy (2) more efficient input clustering mechanisms to handle unusually dispersed data. In so doing, our straightforward proposed model will be an ideal tool for filtering superflorous features of software effort estimation.

REFERENCES

[1] P. Jodpimai, P. Sophatsathit, and C. Lursinsap, “Analysis of Effort Estimation Based on Software Project Models,” Proc. IEEE Symp. Communication and Information Technology, IEEE Press, Sep. 2009, pp. 715-720.

[2] B. W. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[3] L. H. Putnam, “A General Empirical Solution to the Macro Software Sizing and Estimating Problem,” Proc. IEEE Transactions on Software Engineering, IEEE Press, July. 1978, pp. 345-361.

[4] Y. F. Li, M. Xie, and T. N. Goh, “A Study of Genetic Algorithm for Project Selection for Analogy Based Software Cost Estimation,” Proc. IEEE Industrial Engineering and Engineering Management, IEEE Press, Dec. 2007, pp. 1256-1260.

[5] N. H. Chiu and S. J. Huang, “The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances,” Journal of Systems and Software, vol. 80, Apr. 2007, pp. 628-640, doi:10.1016/j.jss.2007.12.793.

[6] S. Jen Huang, N. H. Chiu, and L. W. Chen, “Integration of The Grey Relational Analysis with Genetic Algorithm for Software Effort Estimation,” European Journal of Operational Research, vol. 188, Aug. 2008, pp. 898-909, doi:10.1016/j.ejor.2007.07.002.

[7] M. Lefley and M. J. Shepperd, “Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets,” in LNCS, vol. 2724, E. Cantu-Paz et al., Eds. Springer, 2003, pp. 2477-2487, doi:10.1007/3-540-45110-2.

[8] K. V. Kumar, V. Ravi, M. Carr, and N.R. Kiran, “Software Development Cost Estimation Using Wavelet Neural Networks,” Journal of Systems and Software, vol. 81, Nov. 2008, pp. 1853-1867, doi:10.1016/j.jss.2007.12.793.

[9] N. Tadayon, “Neural Network Approach for Software Cost Estimation,” Proc. IEEE The International Conference on Information Technology: Coding and Computing (ITCC’05), IEEE Press, Apr. 2005, pp. 815-818.

[10] B. T. Rao, “A Novel Neural Network Approach For Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN),” International Journal of Computer Science and Network Security, vol. 9, Jun. 2009, pp. 126-131.

[11] W. Pedrycz, J. F. Peters, and S. Ramanna, “A Fuzzy Set Approach to Cost Estimation of Software Projects,” Proc. IEEE Canadian Conference on Electrical and Computer Engineering Shaw Conference Center, IEEE Press, May. 1999, pp. 1068-1073.

[12] S. J. Huang and N. H. Chiu, “Applying Fuzzy Neural Network to Estimate Software Development Effort,” Applied Intelligence, Springer Press, Apr. 2009, pp. 73-83, doi:10.1007/s10489-007-0097-4.

[13] Z. Chen, T. Menzies, D. Port, and B. Boehm, “Feature Subset Selection Can Improve Software Cost Estimation Accuracy,” Proc. Predictor models in software engineering, ACM Press, May. 2005, pp. 1-6.

[14] S. Koch and J. Mitlohner, “Software Project Effort Estimation with Voting Rules,” Proc. Decision Support System, ACM Press, Mar. 2009, pp. 895-901.

[15] S.J. Huang and N.H. Chiu, “Optimization of Analogy Weights by Genetic Algorithm for Software Effort Estimation,” Information and Software technology, vol. 48, Nov. 2006, pp. 1034-1045, doi:10.1016/j.infsof.2005.12.020.

[16] M. Shepperd and C. Schofield, “Estimating Software Project Effort Using Analogies,” Proc. IEEE Transactions on Software Engineering, IEEE Press, Nov. 1997, pp. 736-743.

[17] NASA60 data set, http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa.arff , accessed on Nov. 5, 2009.

[18] NASA93 data set, http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa_2.arff, accessed on Nov. 5, 2009.

[19] A. J. Albrecht and J. E. Gaffney, “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation,” Proc. IEEE Transactions on Software Engineering, IEEE Press, Nov. 1983, pp. 639-648.

[20] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, “Software Development Cost Estimation Using Function Points” Proc. IEEE Transactions on Software Engineering, IEEE Press, Apr. 1994, pp. 275-287.

[21] A. Abran and P. N. Robillard, “Function Points Analysis: An Empirical Study of Its Measurement Processes” Proc. IEEE Transactions on Software Engineering, IEEE Press, Dec. 1996, pp. 895-910.

[22] Desharnais data set, http://promise.site.uottawa.ca/SERepository/datasets/desharnais.arff, accessed on Nov. 5, 2009.

[23] Stuttgart Neural Network Simulator (Version 4.2), University of Stuttgart and University of Tubingen.

[24] D. Port and M. Korte, “Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research,” Proc. ACM-IEEE international symposium on Empirical software engineering and measurement, ACM Press, Oct. 2008, pp. 51-60.